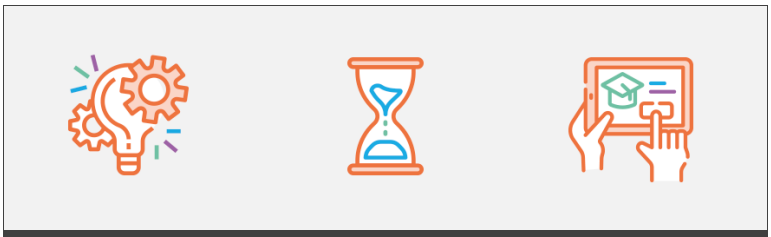
JavaScript (IFCD82)



Área: INFORMÁTICA
Duración: 62h
Metodología: online

Objetivos

Obtener las bases de generación de código en lenguaje JavaScript para el desarrollo e implementación de aplicaciones web, tanto en el lado del cliente como el lado del servidor.

Contenidos y estructura del curso

Introducción
Introducción
Comprensión de los conceptos fundamentales en JavaScript
Desarrollo del código JavaScript sólido y legible
Aprendizaje de las mejores prácticas de desarrollo en JavaScript
Resumen

Conceptos básicos en JavaScript (I): Qué es JavaScript y desplegar entorno de desarrollo
Introducción
Qué es, qué no es JavaScript y qué representa en el desarrollo web actual
Despliegue del entorno de desarrollo, Aptana Studio
Pasos para desplegar el entorno de desarrollo con Aptana Studio
Características de Aptana Studio
Visual Studio Code
Instalar Visual Studio Code
Node.js
Instalar Node.js
Buenas prácticas y recomendaciones

1. Utilizar nombres descriptivos
2. Comentar el código
3. Evitar variables globales
4. Usar camelCase para nombrar variables y funciones
5. Realizar pruebas y depuración
6. Mantener el código limpio y legible
7. Actualizarse constantemente
8. Optimizar la reutilización del código
9. Seguir las recomendaciones de ECMAScript
10. Manejo adecuado de errores

11. Promover la seguridad del código
12. Refactorizar y mantener el código actualizado
    Resumen

Conceptos básicos en JavaScript (II): Tipos en JavaScript
Introducción
Variables. Diferencias entre ámbito local y global. Instrucciones Let y Var
La instrucción var
La instrucción let
Diferencias entre ámbito local y global
Variables y ámbito en bucles y condicionales
Constantes
Tipos primitivos. Tipado dinámico
Tipos primitivos
Tipado dinámico
Objetos
Creación de Objetos
Prototipos y Herencia
Arrays
Creación y Manipulación de Arrays
Arrays Multidimensionales
Características Avanzadas de Arrays
Funciones
Expresiones de función
Arrow functions
Funciones como valores y callbacks
Closures en JavaScript
Parámetros predeterminados y rest
Desestructuración de Argumentos
Resumen

Operadores en JavaScript (I): Aritméticos, de asignación, de comparación y de igualdad
Introducción
Conceptos básicos sobre operadores en JavaScript
Operadores aritméticos
Suma (+)
Resta (-)
Multiplicación (*)
División (/)
Módulo (%)
Operadores unarios
Incremento (++)
Decremento (--)
Exponenciación (**)
Positivo unario (+)
Negación unaria (-)
Precedencia de operadores
Operadores de asignación
Operador de asignación de adición +=
Operador de asignación de sustracción -=
Operador de asignación de multiplicación *=
Operador de asignación de división /=
Operador de asignación de módulo %=
Operador de asignación de exponenciación **=
Operadores de comparación
Operador "Mayor que" (>)
Operador "Menor que" (<)
Operador "Mayor o igual que" (>=)
Operador "Menor o igual que" (<=)
Operadores de igualdad
Operador de Igualdad Abstracta (==)
Operador de Igualdad Estricta (===)
Operador de Desigualdad Abstracta (!=)
Operador de Desigualdad Estricta (!==)
Comparación de cadenas
Resumen

Operadores en JavaScript (II): Ternarios, lógicos y booleanos, bitwise y jerarquización de operadores
Introducción
Operadores ternarios
Uso básico del operador ternario
Ventajas sobre la estructura if-else
Anidación de operadores ternarios
Cuestiones de legibilidad
Buenas prácticas
Operadores lógicos y booleanos
Operador AND (&&)
Operador OR (||)
Operador NOT (!)
Short-circuit evaluation
Operadores BitWise
Operador AND (&)
Operador OR (|)
Operador XOR (^)
Operador NOT (~)
Operadores de desplazamiento (Shift Operators)
Desplazamiento a la izquierda (Left shift)
Desplazamiento a la derecha con signo (Sign-propagating right shift)
Desplazamiento a la derecha con cero (Zero-fill right shift)
Jerarquización de operadores
Orden de Precedencia
Resumen

Control de flujo de ejecución en JavaScript (I): If y switch
Introducción
Sentencia if...else
Sentencia if
Sentencia else
Sentencia else if
Operadores lógicos en condiciones
Mejores prácticas al utilizar if, else...if y else
Sentencia switch...case
Sintaxis básica de la sentencia switch
Case
Break
Default
Grupos de casos y simplificación
Buenas prácticas en el uso de switch
Resumen

Control de flujo de ejecución en JavaScript (II): For y while
Introducción
Sentencia for
Usos avanzados de la sentencia for
Manipulación de arrays con sentencia for
Anidación de ciclos for
Optimizaciones de rendimiento en ciclos for
Sentencia for...in
Manipulación de arrays con sentencia for...in
Considereciones al utilizar for...in
Sentencia for...of
Iterando sobre un array
Iterando sobre un string
Iterando sobre mapas y conjuntos
Uso avanzado del for...of con destructuración
Iterando sobre elementos HTML
Sentencia while
Precauciones al usar bucles while
Manejo de múltiples condiciones
Sentencia do...while
Ejemplo básico de do...while
Precauciones al utilizar do...while
Uso en aplicaciones reales
Comparación con otros bucles

Variaciones en la Condición
Anidamiento de bucles
Refactorización de bucles infinitos
Mejores prácticas
Resumen

Control de flujo de ejecución en JavaScript (III): Break y continue
Introducción
Extra - Cómo evitar bucles infinitos
Validación de condiciones antes de iterar
Establecimiento de un contador de seguridad
Revisión y depuración
Uso de declaraciones condicionales dentro de bucles
Timer o timeout como mecanismo de escape
Pruebas unitarias y tests de integración
Sentencias break and continue
Sentencia break
Definición y utilidad de la sentencia break
Uso de break en bucles
Uso de break en if
Uso de break en switch
Consideraciones al usar break
Sentencia continue
Concepto de la sentencia continue
Uso de la sentencia continue en un bucle for
Uso de la sentencia continue en un bucle while
Consideraciones al usar continue
Prácticas recomendadas y casos de uso comunes
Resumen

Objetos en JavaScript (I): Definición y propiedades
Introducción
Definición de los objetos y de su naturaleza dinámica
Definición de los objetos
Propiedades dinámicas
Notación de corchetes y puntos
Referencias. Tipos
Referencias
Copia por referencia
Comparación por referencia
Entendiendo las referencias en la memoria
Funciones y efectos de lado
Tipos
Tipos primitivos
Tipos de objeto
Referencias en cada tipo
Enumeración de las propiedades de un objeto
Bucle for...in
Método Object.keys()
Método Object.getOwnPropertyNames()
Método Object.entries()
Método Object.values()
Comparación con bucle for
Enumeración de propiedades no enumerables
Resumen

Objetos en JavaScript (II): Uso y clonación
Introducción
Uso de objetos en JavaScript
Prototipos
Inmutabilidad
Clonación de un objeto
Métodos Shallow Clone
Métodos Deep Clone
Bibliotecas para clonación
Patrones de diseño para la clonación
Consideraciones de rendimiento

Buenas prácticas en la clonación de objetos
¡Las funciones son objetos en JavaScript!
Resumen

Objetos en JavaScript (III): Factory y Constructor
Introducción
Funciones Factory
Características de las funciones Factory
Implementación básica de una función Factory
Encapsulamiento con funciones Factory
Ventajas sobre el constructor tradicional
Función Factory con opciones configurables
Uso de funciones Factory para módulos
Limitaciones de las funciones Factory
Funciones Constructor
Definición y uso de funciones Constructor
Añadiendo métodos al prototipo
Patrón Constructor vs. Factory
Heredando propiedades y métodos
Propiedades y métodos estáticos
Privacidad y encapsulamiento
Conclusiones sobre funciones Constructor
Propiedad del constructor
Recolección de basura
Consideraciones de rendimiento y memoria
Estrategias para minimizar la necesidad de recolección
Resumen

Objetos en JavaScript (IV): Math y String
Introducción
El objeto Math
Propiedades del objeto Math
Métodos para operaciones aritméticas
Funciones trigonométricas
Redondeo
Truncamiento
Números aleatorios
El objeto String
Métodos comunes del objeto String
Expresiones regulares y el objeto String
Trabajando con unicode en el objeto String
Conclusiones del objeto String
Resumen

Objetos en JavaScript (V): Plantillas Litertales y objetos Fecha
Introducción
Plantillas literales (plantillas de cadenas)
Definición y sintaxis
Incorporación de expresiones
Cadenas multilínea
Etiquetas en plantillas literales
Uso práctico en desarrollo web
El objeto Fecha
Creación de objetos Fecha
Fecha y hora actual
Fecha específica
Fecha a partir de milisegundos
Obtener componentes de fecha y hora
Establecer componentes de fecha y hora
Formato y conversión de fechas
Operaciones con fechas
Sumar y Restar Días
Trabajar con Meses y Años
Comparar Fechas
Calcular la Diferencia entre Fechas
Aplicaciones prácticas en desarrollo web
Creación de un objeto

Reducción de una matriz con el método reduce()
Resumen

Funciones en JavaScript (I): Expresiones y declaraciones, Hoisting y argumentos
Introducción
Diferencias entre expresiones y declaraciones
Declaraciones de funciones en JavaScript
Expresiones de funciones en JavaScript
Diferencias Clave
Hoisting en JavaScript
Diferencias con 'let' y 'const'
Hoisting de funciones
Funciones expresadas y hoisting
Importancia del orden del código
Buenas prácticas para evitar problemas con el hoisting
Argumentos de una función
Objeto arguments
Resumen

Funciones en JavaScript (II): El operador Rest y parámetros predeterminados
Introducción
El operador Rest
Uso del operador Rest en funciones
Uso del operador Rest en arrays y objetos
Diferencias entre los parámetros Rest y los argumentos
Ejemplo avanzado del operador Rest
Restricciones y buenas prácticas
Parámetros predeterminados
Concepto de parámetros predeterminados
Uso avanzado de parámetros predeterminados
Parámetros predeterminados y tipado dinámico
Evaluación perezosa de parámetros predeterminados
Resumen

Funciones en JavaScript (III): "Getters" y "Setters", gestión de excepciones y this
Introducción
"Getters" y "Setters"
Creación de Getters y Setters
Getters y Setters en Clases
Buenas prácticas y uso de Getters y Setters
Gestión de excepciones con la sentencia try...catch
Funcionamiento y sintaxis
La cláusula finally
Propagación y manejo de errores
Errores personalizados
La palabra clave This
Contexto de this en funciones globales y métodos de objetos
Uso de this en constructores y clases
Manipulación del valor de this con bind, call y apply
this en el contexto de las funciones flecha
this en eventos DOM
Resumen

Metodología

En Criteria creemos que para que la formación e-Learning sea realmente exitosa, tiene que estar basada en contenidos 100% multimedia (imágenes, sonidos, vídeos, etc.) diseñados con criterio pedagógico y soportados en una plataforma que ofrezca recursos de comunicación como chats, foros y conferencias...Esto se logra gracias al trabajo coordinado de nuestro equipo e-Learning integrado por profesionales en pedagogía, diseño multimedia y docentes con mucha experiencia en las diferentes áreas temáticas de nuestro catálogo.

Perfil persona formadora

Esta acción formativa será impartida por un/a experto/a en el área homologado/a por Criteria, en cumplimiento con los procedimientos de calidad, con experiencia y formación pedagógica.

En Criteria queremos estar bien cerca de ti, ayúdanos a hacerlo posible:
¡Suscríbete a nuestro blog y síguenos en redes sociales!

Blog de Criteria